MICROINSTRUCCIONES.

La unidad de control es el ***"cerebro"*** del microprocesador. Esta unidad realiza la decodificación de los códigos de operación y de los códigos de función para poder identificar la instrucción que se va a ejecutar y su tipo (Tipo I, Tipo R y Tipo J). Una vez identificada la instrucción, la unidad de control activa o no cada una de las señales de control de cada unidad funcional (ALU, Archivo de registros, Memoria de datos, Memoria de programa, Contador de Programa) del microprocesador.

De esta manera se forma un código que genera la unidad de control al que llamamos ***microinstrucción***. Por lo tanto, podemos definir a ***una microinstrucción como cada uno de los códigos que genera la unidad de control para activar las señales de control de cada unidad funcional del procesador. Estos códigos se generan en cada ciclo de reloj para poder ejecutar una instrucción del ensamblador***.

Para entender como se realiza la ejecución de las instrucciones a través del procesador y como se generan las microinstrucciones, analicemos los ejemplos siguientes.

Ejemplo 1. Programa que genera un contador. Observe el código mostrado en la tabla 1.

Instrucciones	Dir	WPC	SR1	SWD	WR	SOP	SOP2	ALUOP	LF	WD	SR
		D12	D11	D10	D9	D8	D7	D6D5D4D3	D2	D1	D0
LI R0, #1	0	0	0	0	1	0	0	0000	0	0	0
LI R1, #7	1	0	0	0	1	0	0	0000	0	0	0
CICLO: ADD R1, R1, R0	2	0	0	1	1	0	0	0011	1	0	1
SWI R1, 5	3	0	1	0	0	0	0	0000	0	1	0
B CICLO	4	1	0	0	0	0	0	0000	0	0	0

Tabla 1 Microinstrucciones del programa ejemplo 1.

Durante la ejecución de cada instrucción TODAS las señales de control de cada bloque del microprocesador se encuentran por defecto con cero. Lo primero que se realiza es tomar una instrucción de la memoria de programa, eso quiere decir, que se lee la localidad 0 de la memoria y se saca el primer dato de 15 bits guardado ahí. Después la unidad de control decodifica la instrucción para saber que tipo de instrucción acaba de leer de la memoria de programa. Posteriormente la unidad de control activa las correspondientes señales de control del procesador para ejecutar la instrucción.

La ejecución del programa comienza después de un reset o clear. Esto provoca que el Contador de Programa (PC) comience en cero al igual que todos los registros del procesador. En este ejemplo en particular la primera instrucción a ejecutar es LI R0, #1. Esta instrucción sale de la memoria de programa y el código de operación, colocado en los bits 14…10, se va directamente a la unidad de control. El código del

registro R0, colocado en los bits 9 y 8, se coloca en el bus de entrada WRITE REGISTER del archivo de registros. El número 1, colocado en los bits 7…0, se coloca en el bus de entrada WRITE DATA del archivo de registros. La instrucción se ejecuta cuando la unidad de control activa la señal WR del archivo de registros, esto provoca que en el siguiente flanco de subida de la señal de reloj se cargue el número 1 en el registro R0 y que el valor del PC se incremente en uno. La siguiente instrucción LI R1, #7 se ejecuta de la misma manera.

Después de la ejecución de las instrucciones de carga, la siguiente instrucción a ejecutar se encuentra en la dirección 2 de memoria, la cual es ADD R1, R1, R0. Esta instrucción sale de la memoria de programa y el código de operación, colocado en los bits 14…10, se va directamente a la unidad de control. Como se trata de una instrucción tipo R el código de función, colocado en los bits 3…0, también se va a la unidad de control. El código del registro R1, colocado en los bits 9 y 8, se coloca en el bus de entrada WRITE REGISTER del archivo de registros. El código del registro R1, colocado en los bits 7 y 6, se coloca en el bus de entrada READ REGISTER 1 del archivo de registros. Esto provoca que el número 7 almacenado en R1 pase directamente a la ALU. El código del registro R0, colocado en los bits 5 y 4, se coloca en el bus de entrada READ REGISTER 2 del archivo de registros. Esto provoca que el número 1 almacenado en R0 pase directamente a la ALU. La instrucción se ejecuta cuando la unidad de control coloca el código “0011” en el bus ALUOP, activa la señal WR del archivo de registros, la señal SR, la señal SWD y la señal LF del registro de banderas **(La señal LF se activa con todas las instrucciones que modifican las banderas de la ALU).** Esto provoca que se realice la suma de los operandos de la ALU y que el resultado de la suma se coloque en el bus de entrada WRITE REGISTER del archivo de registros. Entonces, el resultado de la suma se almacena en el registro R1 en el siguiente flanco de subida de la señal de reloj y que el valor del PC se incremente a tres.

Posteriormente, la siguiente instrucción a ejecutar es SWI R1, 5. Esta instrucción sale de la memoria de programa y el código de operación, colocado en los bits 14… 10, se va directamente a la unidad de control. El código del registro R1, colocado en los bits 9 y 8, se coloca en el bus de entrada READ REGISTER 1 del archivo de registros. Esto provoca que el número 8 almacenado en R1 se coloque en el bus, Di7…Di0, de la memoria de datos. El número 5, colocado en los bits 7…0, se coloca en el bus de direcciones A7…A0 de la memoria de datos. La instrucción se ejecuta cuando la unidad de control activa la señal WD de la memoria de datos y la señal SR1**.** Esto provoca que se escriba el número 8 en la dirección de memoria 5 en el siguiente flanco de subida de la señal de reloj y que el valor del PC se incremente a cuatro.

Finalmente, la última instrucción a ejecutar es B CICLO. Esta instrucción sale de la memoria de programa y el código de operación colocado en los bits 14…10 se va directamente a la unidad de control. El código colocado en los bits 9 y 8 no se utiliza en esta instrucción. El número 2, colocado en los bits 7…0, se coloca en el bus de entrada D7…D0 del Contador de Programa. La instrucción se ejecuta cuando la unidad de control activa la señal WPC**.** Esto provoca que se escriba la dirección 2 en PC y se de el salto a la etiqueta CICLO.

Ejemplo 2. Programa que obtiene los primeros 12 términos de la serie de Fibonacci. Observe el código mostrado en la tabla 3.

Instrucciones	Dir	WPC	SR1	SWD	WR	SOP	SOP2	ALUOP	LF	WD	SR
LI R0, #0	0	0	0	0	1	0	0	0000	0	0	0
LI R1, #1	1	0	0	0	1	0	0	0000	0	0	0
LI R2, #0	2	0	0	0	1	0	0	0000	0	0	0
CICLO: ADD R0, R0, R1	3	0	0	1	1	0	0	0011	1	0	1
SWI R0, 72	4	0	1	0	0	0	0	0000	0	1	0
ADD R1, R0, R1	5	0	0	1	1	0	0	0011	1	0	1
SWI R1, 72	6	0	1	0	0	0	0	0000	0	1	0
ADDI R2, R2, #2	7	0	0	1	1	0	1	0011	1	0	1
CPI R2, #12	8	0	1	0	0	1	1	0111	1	0	0
BNEQ CICLO (Si la condición se cumple)	9	1	0	0	0	0	0	0000	0	0	0
BNEQ CICLO (Si la condición no se cumple)	9	0	0	0	0	0	0	0000	0	0	0
FIN: NOP	10	0	0	0	0	0	0	0000	0	0	0
B FIN	11	1	0	0	0	0	0	0000	0	0	0

Tabla 2 Microinstrucciones del programa ejemplo 3.

En este ejemplo analizaremos la instrucción ADDI R2, R2, #2 ubicada en la dirección siete de memoria.

Esta instrucción sale de la memoria de programa y el código de operación, colocado en los bits 14…10, se va directamente a la unidad de control. El código del registro R2, colocado en los bits 9 y 8, se coloca en el bus de entrada WRITE REGISTER del archivo de registros. El código del registro R2, colocado en los bits 7 y 6, se coloca en el bus de entrada READ REGISTER 1 del archivo de registros. Esto provoca que el número almacenado en R2 pase directamente a la ALU. El número inmediato, colocado en los bits 5…0 pasa por el Extensor de Signo para completar los dos bits faltantes. La instrucción se ejecuta cuando la unidad de control coloca el código “0011” en el bus ALUOP, activa la señal WR del archivo de registros, la señal SR, la señal SWD, la señal SOP2 y la señal LF del registro de banderas **(La señal LF se activa con todas las instrucciones que modifican las banderas de la ALU).** Esto provoca que se realice la suma de los operandos de la ALU y que el resultado de la suma se coloque en el bus de entrada WRITE REGISTER del archivo de registros. Entonces, el resultado de la suma se almacena en el registro R2 en el siguiente flanco de subida de la señal de reloj y que el valor del PC se incremente a ocho.

Posteriormente, la siguiente instrucción a ejecutar es CPI R2, #12. Esta instrucción sale de la memoria de programa y el código de operación, colocado en los bits 14…10, se va directamente a la unidad de control. El código del registro R2, colocado en los bits 9 y 8, se coloca en el bus de entrada READ REGISTER 1 del archivo de registros. Esto provoca que el número almacenado en R2 pase directamente a la ALU por READ DATA 1. El número inmediato a comparar, colocado en los bits 7…0, pasa por al operando B de la ALU. La instrucción se ejecuta cuando la unidad de control coloca el código “0111” en el bus ALUOP, la señal SR1, la señal SOP, la señal SOP2 y la señal LF del registro de banderas **(La señal LF se activa con todas las instrucciones que modifican las banderas de la ALU).** Esto provoca que se realice la resta de los operandos de la ALU y que se guarde el resultado de las banderas en el registro de banderas para determinar el tipo de condición de los operandos. Entonces, en el siguiente flanco de subida de la señal de reloj el valor del PC se incrementa a nueve.

Después, la siguiente instrucción a ejecutar es BNEQ CICLO. Esta instrucción sale de la memoria de programa y el código de operación, colocado en los bits 14…10, se va directamente a la unidad de control. La dirección de salto, colocada en los bits 7…0, se colocan en el bus de datos de entrada del PC. En la unidad de control se verifican los valores de las banderas del registro de banderas para determinar la condición de los operandos a comparar. Si la condición se cumple se ejecuta el salto condicional cuando la unidad de control activa la señal WPC. Entonces, en el siguiente flanco de subida de la señal de reloj el valor del PC se carga con tres.

INSTRUCCIONES DE CARGA Y ALMACENAMIENTO								
Instr.	Ejemplo	Significado	Formato de instrucción					Microinstrucción
LI	LI Rd, #Slit8	Rd = Slit8	01	Rd	Slit8			WR
LWI	LWI Rd, lit8	Rd = Mem[lit8]	02	Rd	lit8			SWD WR
SWI	SWI Rd, lit8	Mem[lit8] = Rd	03	Rd	lit8			SR1 WD
INSTRUCCIONES ARITMÉTICAS								
ADD	ADD Rd,Rs1,Rs2	Rd = Rs1+Rs2	00	Rd	Rs1	Rs2	00	SWD WR ALUOP=0011 LF SR
SUB	SUB Rd,Rs1,Rs2	Rd = Rs1-Rs2	00	Rd	Rs1	Rs2	01	SWD WR ALUOP=0111 LF SR
ADDI	ADDI Rd,Rs1,#Slit6	Rd = Rs1+Slit6	04	Rd	Rs1	Slit6		SWD WR SOP2 ALUOP=0011 LF SR
SUBI	SUBI Rd,Rs1,#Slit6	Rd = Rs1-Slit6	05	Rd	Rs1	Slit6		SWD WR SOP2 ALUOP=0111 LF SR
INSTRUCCIONES LÓGICAS								
AND	AND Rd,Rs1,Rs2	Rd = Rs1 & Rs2	00	Rd	Rs1	Rs2	02	SWD WR ALUOP=0000 LF SR
OR	OR Rd,Rs1,Rs2	Rd = Rs1 | Rs2	00	Rd	Rs1	Rs2	03	SWD WR ALUOP=0001 LF SR
XOR	XOR Rd,Rs1,Rs2	Rd = Rs1 ^ Rs2	00	Rd	Rs1	Rs2	04	SWD WR ALUOP=0010 LF SR
NAND	NAND Rd,Rs1,Rs2	Rd = ~(Rs1 & Rs2)	00	Rd	Rs1	Rs2	05	SWD WR ALUOP=1101 LF SR
NOR	NOR Rd,Rs1,Rs2	Rd = ~(Rs1 | Rs2)	00	Rd	Rs1	Rs2	06	SWD WR ALUOP=1100 LF SR
XNOR	NOR Rd,Rs1,Rs2	Rd = ~(Rs1 ^ Rs2)	00	Rd	Rs1	Rs2	07	SWD WR ALUOP=0110 LF SR
INSTRUCCIÓN DE SALTO INCONDICIONAL								
B	B lit8	PC = lit8	06	S/U	lit8			WPC
INSTRUCCIONES DE COMPARACIÓN								
CP	CP Rs1, Rs2	Rs1 – Rs2 Reg_edo = Flags	00	S/U	Rs1	Rs2	08	ALUOP=0111 LF

AUTOR: VICTOR HUGO GARCÍA ORTEGA

CPI	CPI Rd, #Slit8	Rd – Slit8 Reg_edo = Flags	07	Rd	Slit8			SR1 SOP SOP2 ALUOP=0111 LF
INSTRUCCIONES DE SALTOS CONDICIONALES								
BEQ	BEQ lit8	If(EQ) PC = lit8	08	S/U	lit8			WPC (Si se cumple la condición)
BNEQ	BNEQ lit8	If(NEQ) PC = lit8	09	S/U	lit8			WPC (Si se cumple la condición)
BLT	BLT lit8	If(LT) PC = lit8	10	S/U	lit8			WPC (Si se cumple la condición)
BLET	BLET lit8	If(LET) PC = lit8	11	S/U	lit8			WPC (Si se cumple la condición)
BGT	BGT lit8	If(GT) PC = lit8	12	S/U	lit8			WPC (Si se cumple la condición)
BGET	BGET lit8	If(GET) PC = lit8	13	S/U	lit8			WPC (Si se cumple la condición)
OTRAS INSTRUCCIONES								
NOP	NOP	No operation	14	S/U	S/U	S/U	S/U	

AUTOR: VICTOR HUGO GARCÍA ORTEGA